FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

3D Model-Based Recognition

Maria Ausenda Miranda

WORKING VERSION

Mestrado Integrado em Engenharia Electrotécnica e de Computadores

Supervisor: Luís Filipe Teixeira

Second Supervisor: Manuel João Ferreira

July 26, 2016

Resumo

O objetivo deste projeto foi desenvolver um estudo comparativo de algoritmos de reconhecimento de objetos 3D usando o modelo base, constituído por uma fase on-line e outra off-line, e assim, selecionar um possível método a ser incorporado nas ferramentas da Enermeter, uma empresa local que propôs este trabalho.

É essencial que um sistema baseado em visão por computador seja implementado com técnicas de aquisição e processamento de imagem rápidas e eficazes, de modo a actuarem num curto espaço de tempo, como é o caso de robôs autónomos e veículos de condução autónoma. Para tal, é necessário criar previamente uma base de dados com as representações dos modelos apropriados para uma dada aplicação, restando apenas o processamento e reconhecimento da imagem adquirida em tempo real.

Assim, foram implementadas duas soluções baseadas em algoritmos já existentes nesta área científica. A primeira consiste na construção de pares de pontos característicos (PPF) que representam a nuvem de pontos a um nível global. Contudo, o reconhecimento é feito localmente, agrupando e alinhando características semelhantes. A alternativa reside na criação de um descritor baseado no tradicional algoritmo de *spin-images*, em que são criadas três características (*TriSI feature*) que contêm a informação geométrica envolvente a um determinado ponto. Este processo é repetido ao longo de toda a nuvem, finalizando com a verificação individual de cada modelo, de modo a selecionar o que melhor caracteriza a imagem capturada.

Finalmente, de modo a avaliar as referidas propostas, foram realizados vários testes em bases de dados com objectos comuns do quotidiano e um conjunto de modelos relativos ao ambiente industrial onde a Enermeter se concentra. Os resultados foram satisfatórios na medida em que a comparação dos algoritmos foi conseguida e foi selecionada uma estratégia possível de ser acrescentada às já existentes técnicas baseadas em visão da Enermeter.

Abstract

The goal of this project was to develop a comparative study of 3D object recognition algorithms using the model-base, which is constituted by an on-line and an off-line phase, and thus selects a possible method to be incorporated into the Enermeter's tools, a local company that proposed this project.

It is essential that a computer vision based system to be implemented with quick and efficient techniques of image acquisition and processing, to make an action in a short period, as in the case of the autonomous robots and in self-driving vehicles. To do so, it is necessary to create a database previously, with the appropriate models representation to the application, leaving just the processing and the recognition of the image acquired in real-time.

Therefore, it was implemented two solutions based on existent algorithms in this scientific field. The first one consists in construct a point pair feature which represents the point cloud globally. However, the recognition is achieved locally, grouping and aligning similar characteristics. The alternative approach creates a descriptor based on the traditional spin-image feature, which generates three features that encode the geometric information around a point. This process is repeated over the entire point cloud, and each model is verified individually to select the one that best characterize the captured image.

Finally, to evaluate the referred proposals, several tests were implemented in databases with common objects and a group of models related to the industrial environment in which the Enermeter focuses. The results are satisfactory since the comparison between the two algorithms was achieved and was selected a strategy capable of being added to the existing Enermeter's tools.

Agradecimentos

Em especial, ao meu orientador Professor Luís Teixeira por tornar isto possível, pela amizade, preocupação e dedicação. Por me acompanhar desde o primeiro ao último ano e por se ter tornado numa referência.

Ao Professor Manuel João e ao Tiago Pereira da Enermeter por toda ajuda e disponibilidade ao longo deste projeto.

Aos meus pais e irmão por todo o suporte emocional, ajuda e por serem os melhores, sem eles nada disto seria possível.

Ao Francisco pela pessoa incrível que é, por todo o carinho, paciência, por estar sempre presente e acreditar em mim.

Aos meus tios e ao Bernardo, por me fazerem sentir em casa e me acompanharem ao longo destes anos.

À Sofia Inácio, ao André Ferreira e aos fofinhos pela amizade, motivação e pelas gargalhadas intermináveis.

Às minhas meninas, Joana, Rebeca, Marta e Sara que mesmo longe, estão sempre muito perto.

"Tudo é ousado para quem nada se atreve"

Fernando Pessoa

Contents

List of Figures

List of Tables

Abbreviations

1D	One-Dimensional
2D	Two-Dimensional
3D	Three-Dimensional
4D	Four-Dimensional
API	Application Programming Interface
CNNs	Convolutional Neural Networks
HKS	Heat Kernel Signature
ICP	Iterative Closest Point
ISS	Intrinsic Shape Signatures
LRF	Local Reference Frame
NN	Nearest Neighbour
NNDR	Nearest Neighbour Distance Ratio
PCA	Principal Component Analysis
PPF	Point Pair Feature
RANSAC	Random Sample Consensus
RoPS	Rotation Projection Statistics
RR	Recognition Rate
SI	Spin-Image
SURF	Speeded Up Robust Feature
TriSI	Tri-Spin-Image
λ_x	Eigenvalue
e_x	Eigenvector
δ	Surface Variation

Chapter 1

Introduction

1.1 Context

The recognition of a 3D object is part of the artificial vision field. The technological advance of this area has grown exponentially in the last years, especially in sectors where quality and precision play a significant role, as is the case of autonomous robots and self-driving vehicles.

The 3D data is becoming more affordable, due to the recent development of 3D imaging sensors and the real-time simultaneous localization and mapping techniques. This advancement together with the progress in high-speed computers makes the study of 3D environments widely accepted between the computer vision researchers.

Enermeter[1], the company that proposed this project, develops artificial vision systems that can inspect and optimize quality standards. The software developed by *Enermeter* focuses principally on dimensional and position analysis, identification and classification of defects.

1.2 Motivation

The image recognition by humans is performed effortlessly. This feature remains when the object's appearance varies, for instance, the size difference, the point of view of observation, object rotation and even when it is partially obstructed. In the case of systems based on computer vision, this task is not trivial.

In recent decades, the development of algorithms that allow robots to be able to detect and recognize objects quickly have been extensively studied. In cases where it is necessary for a robot to find and handle objects, it is essential to create a system capable of obtaining information about the shape and location of the subject matter in a short period, making possible the use of this technology.

The object recognition is divided mainly into two categories: identification and characterization (Elgammal, 2010). The identification involves distinguishing different objects in the image,

[1] http://www.enermeter.pt/

while the characterization involves grouping a set of different objects that have the same properties. The distinction is achieved by finding a group of points for comparing the analyzed object, using a predefined algorithm.

3D model-based recognition has an off-line and on-line phase, as shown in Figure 1.1. Models of objects are built in the off-line mode from the acquisition of images. To capture the entire object it is necessary to acquire images from multiple views. Then the representation of the model is saved in a library. During on-line recognition process, an image of a scene is converted into a representation. The object is recognized matching the representation with the models of the databases (Mian et al., 2006).

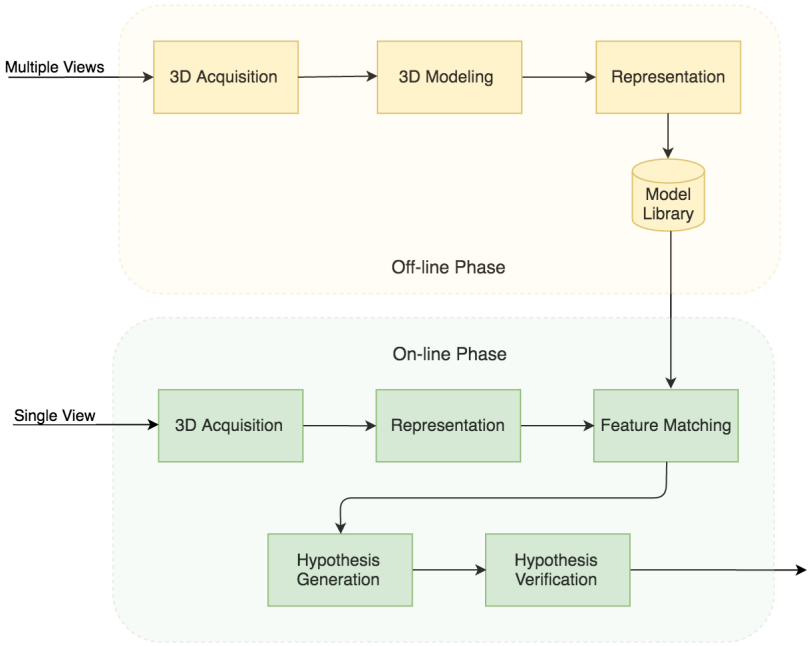

Figure 1.1: Block diagram of the 3D model-based recognition.

1.3 Objectives

The primary goal of this work is to present a comparative study of the existing 3D object recognition strategies, and this way, evaluate the best method to incorporate the Enermeter's software. To recognize an object it is necessary to implement an algorithm capable of making the correspondence between an input image and its model, present in the library, as shown in Figure 1.1.

The first major phase is to study the different approaches about algorithms of feature detection, feature matching and object recognition. In order to select some strategies to develop a detailed evaluation, it is essential to be aware of the recent advancements in this area.

After the study of the related work, the selected algorithms will be implemented. First, the feature points should be detected according to their distinctiveness. Otherwise, the recognition may not be accurate. The next phase consists of extracting geometric information of the local

surface from the feature point and constructs the feature descriptor. Then, after having both the feature descriptors of the input image and all models in the library, it is required to match them.

To actually recognize an object, the descriptor that has the best match is hypothesized to be present in the input image. This hypothesis is verified by aligning the model to the scene. If the alignment is accurate, that object is recognized.

Finally, after the implementation of the methods will produce several tests in artificial and industrial datasets, which will evaluate the results of the algorithms. This study will provide useful information having in consideration the appropriateness of the algorithms in the main purpose of this work.

1.4 Contributions

The major contribution of this thesis was the implementation of a recognition algorithm using the TriSI feature, in C++. This algorithm is compatible with the Point Cloud Library (PCL) and it will be available open-source. The data input can be representations of 3D point clouds or meshes and the parameters can be easily changed, which facilitate the use of it without having a close look at the implemented code.

Another contribution is that this thesis also presents a comparative study of recognition algorithms implemented on two public datasets and on a dataset provided by Enermeter. This is significant since the TriSI based algorithm were not previously tested in real datasets.

1.5 Document Structure

In this chapter was described the scope of the work, as well as the motivation and the project overview. The rest of this thesis will be organized as follows: Chapter 2 contains the related work developed in this area. In Section 2.1 will show a small background about the methods of representation and in Section 2.2 will be described strategies for feature detection and feature description.

Next, in Chapter 3, the characteristics of the method proposed by Drost et al. (2010) are going to be exposed, along with some considerations during the phase of implementation. Chapter 4 will analyze in detail the algorithm proposed by Guo et al. (2015) and the effect of its parameters in the final results.

In Chapter 5, the results of the application of the algorithms in real and synthetic databases will be described.

Finally, in Chapter 6, the conclusions of the comparative study will be presented having in mind the results showed in the previous chapter. It also discusses the future directions about this topic.

Chapter 2

Literature Review

In this chapter, the related work around the problem of 3D recognition is presented. Some of the algorithms mentioned below are partially or entirely used in this thesis. This literature review is based on existing surveys (Guo et al. (2014); Pope (2004); Campbell and Flynn (2001)) and other documents properly referenced throughout the chapter.

2.1 Representation

Concerning that the 3D modeling process was already done in previous work, the objects are presented in the form of point clouds and/or polygonal meshes.

The representation, process of selecting feature points and create feature descriptors, is the most important phase of any recognition algorithm and must have the following principles (Mian et al., 2005):

- Unique
- Represent free-form objects
- Represent occluded and cluttered objects
- Stable and robust to noise
- Efficient regarding speed

Finding a method that achieves all these criteria is a challenging task. For this reason, the majority of techniques only accomplish one of them. The techniques that can manage occlusion and clutter will have more emphasis in the state of the art since it is easy to find an object partially occluded by others.

2.1.1 Feature Detection

In order to have an accurate representation of an object, the process of feature detection must select distinctiveness features. Depending on whether the scale is predefined or can be adapted,

the feature detection can be divided into two components: fixed-scaled detection or adaptive-scale detection.

Zhong (2009) suggested a recognition algorithm, where the feature points are detected with a fixed-scaled approach. The intrinsic shape signatures (ISS) method selects the points based on the surface variations. To do that, Zhong (2009) calculates the weight scatter matrix of the neighboring points of the point p. Then, the points which satisfy these restrictions are possible candidates to be feature points:

$$\frac{\lambda_2}{\lambda_1} < \tau_{21} \qquad \frac{\lambda_3}{\lambda_2} < \tau_{32} \tag{2.1}$$

Lastly, the features points are detected based on the smallest eigenvalue λ_1 (Guo et al., 2014).

Sipiran and Bustos (2011) also introduced a fixed scaled method called Harris 3D detector, which is an extension of Harris detector for 2D images to 3D applications. Initially, the neighboring points, centered in the feature to be analyzed, are translated in order to make the centroid the origin of the 3D coordinate system. Then, the neighborhood is rotated to align the normal of the feature with the z-axis. The method applies the principal component analysis (PCA) to do so and choose the eigenvector with the lowest associated eigenvalue as the normal of the fitting plane.

Finally, this method uses two approaches to select the set of interest points. Figure 2.1 a) is the model representation and figure 2.1 b) represents the selected points with highest Harris response, in this case, only the points with higher salience were obtained. The other approach is represented in figure 2.1 c), where the points are selected by clustering. In this case, the points of interest are well distributed along the object surface.

Figure 2.1: The result of the two options to select interest points in Harris 3D method (Sipiran and Bustos, 2011).

Ho and Gibbins (2008) and Unnikrishnan and Hebert (2008) developed methods of adaptive-scale feature detection. These methods are capable of detecting features by calculating the *maxima* of the surface variations at a set of varying neighborhood sizes. The comparison between these two models is represented in figure 2.2.

The method proposed by Ho and Gibbins (2008) uses local shape variation to extract a multi-scale feature. The variation of the surface is measured by using the standard deviation of the shape index values. Then, they calculated a local variation of the shape index for each scale. Finally, a set of feature points is obtained by finding the set of *maxima* from the scale-space representation. This method is efficient and robust to minor noise. Later, to improve the algorithm in high noisy

surfaces, Ho and Gibbins (2008) obtained the feature points by estimating the curvature at different scales (Guo et al., 2014).

On the other hand, Unnikrishnan and Hebert (2008) used the integral operator $B(p,\rho)$ to capture surface variation at a point p and the neighborhood size is represented as ρ. The integral operator causes the move of point along its normal direction n. This displacement is proportional to the men curvature and, then, the surface variation is defined as:

$$\delta(p,\rho) = \frac{2\|p - B(p,\rho)\|}{\rho}^{-\frac{2\|p-B(p,\rho)\|}{\rho}} \tag{2.2}$$

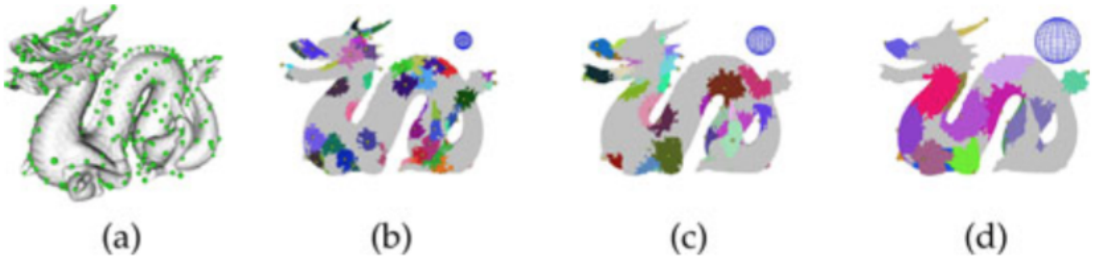

Figure 2.2: Feature detection by Ho and Gibbins (2008) method and by Unnikrishnan and Hebert (2008), using three different scales. Extracted from Guo et al. (2014).

In figure 2.2 b), c) and d) it illustrated the features detected at three different scales. The blue spheres represent the scale of the feature points and each colored area corresponds to a neighborhood (Guo et al., 2014).

2.1.2 Feature Description

After feature detection is completed, it is necessary to extract geometric information of the local surface from the feature point and encoded into a feature descriptor.

Johnson and Hebert (1998) proposed a recognition algorithm titled spin image (SI) representation. They required that the models of the objects are represented in a polygonal regular mesh, which ensures the approximation between their spin image and the image from the real object surface. This method is robust to occlusion and clutter, but results in multiple ambiguous representations, due to mapping a 3D surface to a 2D histogram, making the spin image representation non-unique. Another negative point is the fact that their representation is sensitive to the resolution of the object (Mian et al., 2006).

The point signatures representation is also a technique that represents objects in the presence of occlusion and clutter. Chua and Jarvis (1997) proposal extracts 1D signatures from the surface, rather than encoding information about the points of interest. The contour of the surface is obtained by intersecting the surface with a sphere of predefined radius, which the center coincides with the key-point. Then, they defined a plane to these contour points and translated in the direction of the center of the sphere.

> "The starting point of this signature is defined by the point on the signature that gives the maximum distance from the 3D curve. Point signatures are calculated for every point on the object's surface." Mian et al. (2005)

Therefore, the biggest challenge in point signatures is the definition of the starting point of the signature. If more than one point is a candidate to be a starting point, this technique may be ambiguous and obtain multiple representations from the same point. Point signatures representation is also sensitive to noise and varying mesh resolutions.

Mian et al. (2006) proposed a representation technique based on spin image representation that can handle with occlusion and clutter. They used a multidimensional table representation, called tensors. Figure 2.3 shows the six phases of this algorithm. First, it is created a point could of the object and it is converted into a triangular mesh, where the number of vertices is equal to the number of points in Fig. 2.3 a). Then, they simplify the mesh, reducing the number of vertices. Next, they defined a 3D grid centered in the origin of the coordinate system defined by two points of the mesh. Finally, the tensor value is the corresponding bin of the grid intersecting by the surface area. The algorithm uses a 4D hash table for matching the tensors of a view with those of the remaining views (Mian et al., 2006).

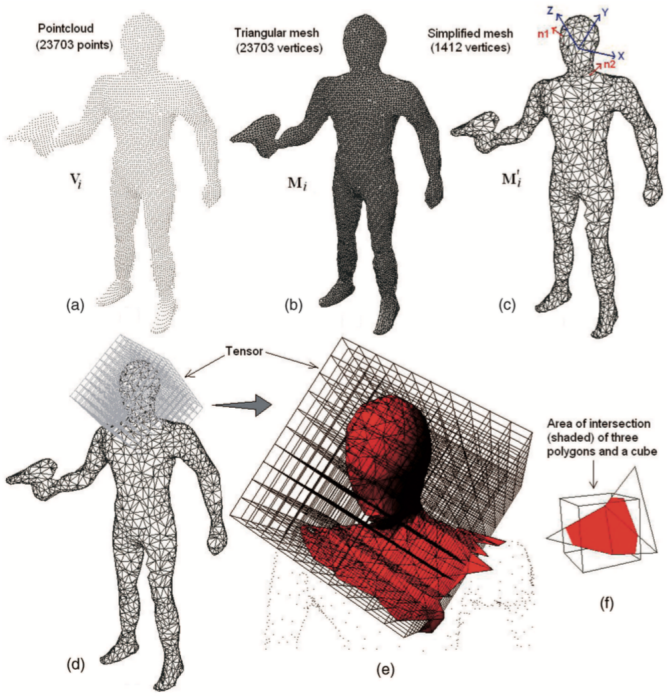

Figure 2.3: Illustration of the tensor representation, including the off-line 3D modeling and the on-line recognition and segmentation phases (Mian et al., 2006).

The ISS method (Zhong, 2009) determines the feature descriptors using a histogram based strategy. Firstly, it is constructed a local reference frame (LRF) by calculating the eigenvectors of the covariance matrix of the neighboring points. Then the spherical angle space is divided into uniformly distributed cells using a 3D partition. The ISS descriptor was constructed by summing

the density weights of all points that fell into each cell. This technique has better results than spin images in the presence of noise, clutter and occlusion (Guo et al., 2014).

Drost et al. (2010) proposed a model representation based on point pairs features (PPF) that describe the relative position and orientation of two oriented points. This feature comprises four components: the distance between two points, the angles between the normals and the vectors of the two points and the angle between the two normals. After that, the model is created by grouping together similar point pairs features of all model. To do that, it is necessary to calculate the PPF for all point pairs. Then, a hash table stores the similar features in the same position and may be accessed by using the $\mathbf{F_s}(\mathbf{s_i}, \mathbf{s_j})$, which $\mathbf{F_s}$ represents the point pair feature of the scene $\mathbf{s}$.

The Point Pair Feature method have satisfactory results in real and synthetic data in the presence of noise, clutter and occlusion.

Later, Guo et al. (2015) proposed, as well, an algorithm based on the spin image representation. They construct a unique and repeatable LRF for all feature points. Then, it was generated three signatures that contain the geometrical information. Finally, the tri-spin-image (TriSI) feature is created by concatenating and compressing the three signatures. This method outperforms the spin image one and it is robust to noise, occlusion, clutter and the mesh resolution variation.

The transform based methods consist in transform a range image from one space domain to another one. Consequently, the 3D neighbourhood is described by encoding the information in that new domain. Hu and Hua (2009) used the *Laplace-Beltrami* spectrum to do so. However, most of the local spatial regions are open boundary surface. To solve this issue, they attach another surface to the open boundary surface with the same shape of the original but with an opposite normal. Thereby, the local descriptor of each feature point is invariant to transformation, isometric deformation and scaling.

Another method based on transform is the heat kernel signature (HKS) proposed by Sun et al. (2009). They restrict the original heat kernel to the temporal domain. The HKS can be interpreted as a multi-scale notion of the Gaussian curvature (Guo et al., 2014). Later, other authors developed versions of HSK capable of being applied in scale invariant and non-rigid shape retrieval.

A version of speeded up robust feature (SURF) for 3D applications developed by Knopp et al. (2010), can be divided into these steps (Guo et al., 2014): Firstly, it is necessary to transform the mesh into a 3D voxel image and apply the *Haar* wavelet transform. Then, for each feature point, it is needed to calculate the local reference frame. The neighbourhood is divided into $N_b \times N_b \times N_b$ bins and the vector v is calculate, for each bin, doing:

$$v = (\sum d_x, \sum d_y, \sum d_z) \tag{2.3}$$

Finally, the feature descriptor is created by combine all vectors.

More recently, Qi et al. (2016) improved the existing 3D representation algorithms based on convolutional neural networks (CNNs), more specifically the volumetric CNNs and multi-view CNNs. These strategies are trained with different representations of a 3D image, shown in figure 2.4. The volumetric representation uses a 3D tensor of binary or real values to codify an image

and the multi-view representation encodes an object as a collection of renderings from multiple viewpoints.

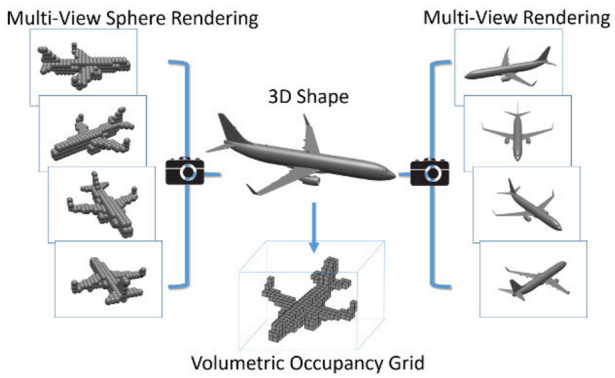

Figure 2.4: Representations of a 3D image using in a CNNs approach (Qi et al., 2016).

Qi et al. (2016) also introduce an extensive experimental evaluation of the traditional CNNs methods and their methods, using different datasets. Their research provides two new architectures of volumetric CNNs that can minimize the performance gap between this one and the multi-view CNNs.

2.2 Recognition

Recognition consists of finding a match between a model and an image. However, there are several challenges associated with this procedure, for instance, the lack of knowledge about which objects might be in the image, their poses and the presence of occlusion and clutter.

After the representation phase, most methods execute the feature matching and the hypothesis verification to recognize an object in an image.

Taking in consideration that recognition is an on-line process, represented in figure 1.1, representation and matching must be fast in order to minimize the recognition time. If the representation process was appropriate, the matching task is easily obtained.

2.2.1 Feature Matching

Feature matching is the process of establishing the correspondence between the model and the image, using the feature descriptors. Feature matching strategies can be separated into three different categories: threshold-based matching, nearest neighbor (NN) based matching and nearest neighbor distance ratio (NNDR) based matching. All of this strategies compare each of the model feature descriptors with all of the input image feature descriptors.

In the case of threshold-based matching, if the distance between two descriptors is less than a threshold τ, then the regions are matched. For this method of matching, a descriptor can have several matches and most of them may be correct.

$$\|F_x - F_y\| < \tau \tag{2.4}$$

On the order hand, in the NN strategy, two regions A and B are matched, if the feature descriptor F_{dB} is the nearest neighbor of the feature descriptor F_{dA} and the distance between them is less than a determinate threshold, which makes only a match per descriptor.

Lastly, the NNDR strategy is identical to NN, but it takes into consideration the first and second nearest neighbor of the feature descriptor. To match the regions, the distance ratio between the first neighbor F_{dB} and the second neighbor F_{dC} need to be less than the threshold.

$$\frac{\|F_{dA} - F_{dB}\|}{\|F_{dA} - F_{dC}\|} < \tau \tag{2.5}$$

Since the NN and the NNDR elect only a descriptor that has the best match and rejects all others, these methods have a higher precision comparing with the threshold-based method. However, as a result of the image transformations, the distance between two similar descriptors differs remarkably Mikolajczyk and Schmid (2005).

Another important factor to be taken into consideration is the search over the model descriptors. A simple procedure compares the feature descriptor of the image with all of the model descriptors but, in a large database, this will have a computational complexity of $O(N)$ (Guo et al., 2014). A better and faster way for searching is to use a convenient data structure or indexing method.

Taking some of the algorithms addressed in this literature review as an example, the spin images recognition method (Johnson and Hebert, 1998) used a slicing based algorithm for search, the point signatures (Chua and Jarvis, 1997) used a 2D index table, the 3D Tensor (Mian et al., 2006) and the PPF (Drost et al., 2010) methods used a hash table, the ISS algorithm (Zhong, 2009) used a locality sensitive tree and the TriSI recognition strategy (Guo et al., 2015) used k-d tree.

2.2.2 Hypothesis Generation and Verification

The last stage of any recognition algorithms is the achievement of a precise transformation hypothesis, with the purpose of rejecting the false matches obtained in feature matching.

The techniques for generating a hypothesis can be separated into a number of different categories (Guo et al., 2014):

- **Geometric Consistency**
  The task of verifying if different correspondences have similar distance and similar relative orientation. This results in a group of correspondences geometrically identical and it is used to calculate the transformation hypothesis. As is the case of the Spin-Images algorithm (Johnson and Hebert, 1998);

- **Pose Clustering**
  A method which consists of finding groups of transformations, by calculating a transformation for each match of feature descriptors. The transformation hypotheses are the centers of the clusters. For example, the ISS method (Zhong, 2009), the PPF one (Drost et al., 2010) and the TriSI method (Guo et al., 2015);

- **Constrained Interpretation Tree**
  The goal is to create an interpretation tree, where the root connect different sub-trees for each model. Each node in a sub-tree contains a hypothesis; that is formed by feature correspondences between the model feature descriptor and the image descriptor;

- **RANSAC**
  A technique that selects a minimum number of feature matches to calculate a transformation. This transformation will be the alignment of the model and the image and will be counted the number of *inliers*, which are the matches that can fit the model. The transformation with the higher number of *inliers* is the transformation hypothesis;

- **Game Theory**
  The features that match will compete in a non-cooperative game. The game will generate a process of exclusion, which the false matches will be rejected. The transformation hypotheses are obtained using the persistent correspondences;

- **Generalized Hough Transform**
  Each match of a model feature descriptor and an image feature descriptor is a point in the Hough space. The transformation hypotheses are the peaks in the Hough accumulator;

- **Geometric Hashing**
  An algorithm that selects a reference coordinate system and uses a hash table to save all model points that have a match, which their coordinates are obtained having in consideration that reference. This method uses geometric invariants to vote for transformation hypotheses.

The transformation hypotheses can be verified globally or individually, in order to reject the wrong assumptions from the correct ones.

The individual verification method consists in aligning a nominee model with the input image using only one transformation hypothesis; then it will be using a method like the iterative closest point (ICP), to refined the alignment. The hypothesis is validated, whenever the accuracy of alignment is superior to a threshold (Guo et al., 2014).

Conversely, the global method of verification handles with the entire group of hypotheses to get an optimal global transformation. Generally, this strategy is less used comparing with the individual one.

2.3 Summary

In this chapter, the problem of the 3D object recognition was studied. Firstly, several methods of feature detection and feature description were exposed, to represent an object accurately. Then, in the recognition phase, the strategies of feature matching, hypothesis generation and verification were presented.

After being conscious of the recent progress in this field, it was selected two algorithms: one based on point pair features and other based on the tri-spin-images features. As implementing methods compatible with the Point Cloud Library (PCL) is a challenging task, firstly, to reduce the initial impact, it was chosen a conventional algorithm available in this library, as is the case of the PPF one. Then, it was searched for a modern strategy to be entirely implemented in the PCL, and the TriSI strategy was the one elected. Another criterion used to select both algorithms is that they must be robust to noise and occlusion.

Chapter 3

Case Study 1: Recognition Algorithm Based On Point Pair Features

As referred in section 2.1, the point pair feature is an algorithm for detecting features with high capabilities of discrimination, but without having the need of a dense point cloud. To do so, in this method the feature descriptor was created globally, where similar features are grouped together, and the recognition is done locally, with a quick voting scheme.

The implementation was made using the point cloud library (PCL[1]) and the Eigen[2] library. PCL is an open-source resource for 2D and 3D applications and presents an extensive approach to the subject of 3D perception. Eigen is a C++ template library for linear algebra and the PCL also uses it.

The representation and the recognition phases were entirely implemented using the code available in the PCL. In an initial phase and to reduce the computational cost, the calculation of the descriptor was obtained with a few models of the database.

3.1 Point Cloud Sub-sampling

The point clouds can be generated by 3D scanners and 3D imaging. The result is a representation of the object's surface. The point clouds are constituted by a group of points measured by these devices. The number of points present in a cloud relies on considerable aspects like the devices used to create the cloud, the size and the content of the image. There are statistics about the number of points, but it is not possible to precise this amount.

Normally, a point cloud can have millions or billions of points and in some cases, it is impossible to process all these points. For instance, in this algorithm, it is necessary to calculate a normal and a feature to each point, which leads to a massive computational cost.

Therefore, the first logic approach is to reduce the points in a cloud, without losing the meaning of it. For that reason, it was selected a filter that fulfills these requirements.

[1] http://pointclouds.org/
[2] http://eigen.tuxfamily.org/

3.1.1 Voxel Grid

Voxel Grid is a C++ class present in the PCL library that, in a simple way, creates minuscules boxes all over the input cloud, with the purpose of filtering and reduce the points on it.

The process of filtering consists in finding the points that fit in a voxel grid, i.e. the minuscules boxes, and, for all the points in it, calculate the centroid. The output cloud has as much points as the number of voxel grids, and each point of it corresponds to the centroid of all points in a particular cell.

In figure 3.1 the original cloud is shown, which has 35 947 points, and figure 3.2 exposes the cloud after applying the voxel grid filter, that results in a cloud of 1 530 points and the underlying surface is accurate with the previous figure. This method was the adopted one, since the results are the most suitable and there is no loss of information during this process.

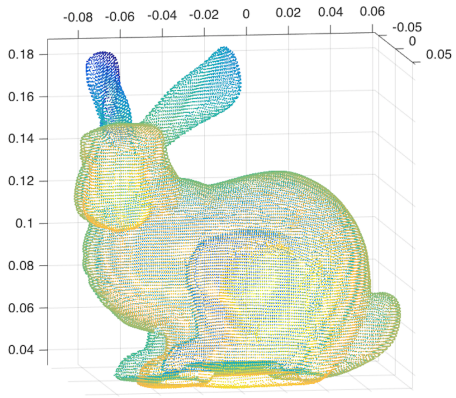

Figure 3.1: Point Cloud of Stanford Bunny before sub-sampling.

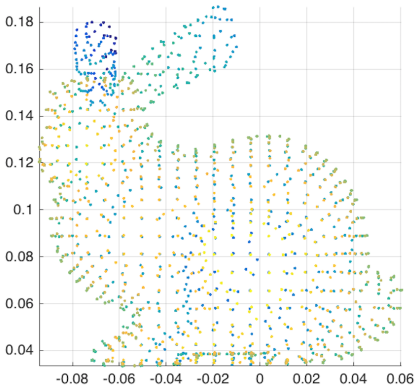

Figure 3.2: Point Cloud of Stanford Bunny after Voxel Grid sub-sampling.

3.1.2 Approximate Voxel Grid

This technique of filtering is also included in the PCL library and is closely identical to the previous one. The notorious difference is the way that the points are approximated. In this case, is the center of the voxel that defines the approximation and not the centroid of all the points in the voxel.

As it is shown in figure 3.3, this method is not the most suitable for the problem of recognition, since the surface contour does not remain reliable.

3.1.3 Grid Minimum

Another method capable of sub-sampling a cloud is the Grade Minimum and it is also based on the Voxel Grid one.

After the determination of the points to be included in a voxel, it starts searching for the point with the z minimum value. Then, the corresponding point in the output cloud will be the searched point. Depending on the size of the cloud and the grid, this technique may not be the most appropriated. In figure 3.4 it is shown the result of it, using a grid of 0.8 cm.

3.1.4 Local Maximum

The last tested strategy to down-sample a point cloud is the Local Maximum and it consists in eliminating the maxima of the points within in a certain radius.

For each point of the cloud, this process finds all the nearest neighbors in a distance range, using the radius search method present in the *KdTreeFLANN* class in PCL library, which is a type of a binary search tree used for organizing a group of points in a space with k dimensions. To determinate the local maximum it is necessary to compare the z dimension of the points.

Figure 3.5 shows the Stanford Bunny down-sampled using a radius of 1 cm, which results in a cloud of 7 078 points. The cloud presented in the figure have an accurate underlying surface, but the points are not distributed uniformly.

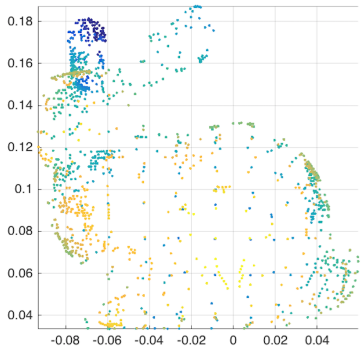

Figure 3.3: Point Cloud of Stanford Bunny after Approximate Voxel Grid sub-sampling.

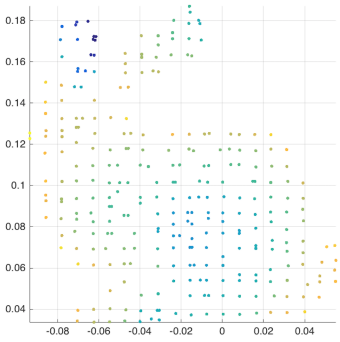

Figure 3.4: Point Cloud of Stanford Bunny after Grid Minimum sub-sampling.

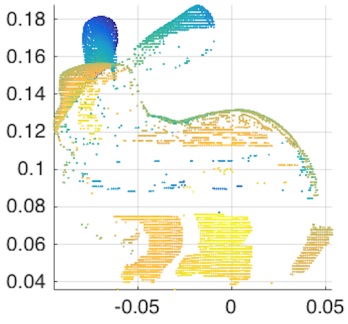

Figure 3.5: Point Cloud of Stanford Bunny after Local Maximum sub-sampling.

3.2 Normal Estimation

As mentioned in section 2.1, the point pair feature describes the relative position and orientation of the two points. Therefore, to have a model represented as a set of oriented points, it is essential to associate a normal with each point of a cloud or a mesh.

The normal vector of a point is equivalent to determining the normal of a plane tangent to the surface at that point. Thus, the proposed solution is to analyze the eigenvalues and the eigenvectors of the covariance matrix created from the nearest neighbors of the point of interest. The normal is obtained with the aid of the principal component analyses (PCA), which is a mathematical procedure that uses an orthogonal transformation to modify a group of observations of correlated variables, towards a set of values of linearly uncorrelated variables.

Considering that for a given point p, all k points within a distance of r from p are the nearest neighbors points q_i. So, the covariance matrix C is given by:

$$C = \frac{1}{k}\sum_{i=1}^{k}(q_i - p)(q_i - p)^T \tag{3.1}$$

To find the eigenvalues and the eigenvectors of the covariance matrix C the following equation is applied, where V is the matrix of the eigenvectors and D is the diagonal matrix of the eigenvalues.

$$V^{-1}CV = D \tag{3.2}$$

Thereby, the j-th eigenvalue of the covariance matrix C is obtained doing $D[j,j] = \lambda_j$ and the corresponding eigenvector is the j-th column of the matrix V. The columns of the D and the V matrix are in descending order of eigenvalues.

Finally, the normal vector is the eigenvector that coincides with the smallest eigenvalue. If the dimensions of the matrix V are 3×3, the normal vector $n = (n_x, n_y, n_z)$ can be extracted as:

$$n = \begin{cases} n_x = V[1,3] \\ n_y = V[2,3] \\ n_z = V[3,3] \end{cases} \tag{3.3}$$

The approach specified uses all the nearest neighbors points inside a sphere with a predefined radius r, but instead of a sphere, it could be utilized the nearest K points. The precision of the result depends essentially on the values of r or K and need to be selected accordingly to the detail level that is desired. The small values are characterized by obtaining a normal vector strictly perpendicular to the surface and the tiny details are evident. In a large scale, the details are omitted and the normal vector is not precisely orthogonal to the surface. The radius used in this work was 3 cm.

This methodology was implemented with the OpenMP[3] API, which is used for multiprocess programming of multi-platform shared memory and it is intended to speed up the computation.

3.3 Off-line Phase

The off-line phase is characterized for training the models and creating the databases, using a suitable representation. The block diagram of the off-line phase of this particular method is exposed in figure 3.6. The first steps consist of preparing the point cloud to be trained. These procedures are the sub-sampling and the normal estimation, section 3.1 and section 3.2 explained them in detail.

Following the diagram, the next step is the model representation, divided into two phases: the calculation of the point pair feature F and the calculation of the model rotation angle α_m.

As referred in section 2.1, the point pair feature is constructed using two points in the model and their corresponding normals. So, for two points m_1 and m_2 with normals n_1 and n_2, respectively, the feature vector F is calculated doing:

$$F(m_1, m_2) = \Big(\|d\|, \angle(n_1, d), \angle(n_2, d), \angle(n_1, n_2) \Big) \tag{3.4}$$

[3] http://openmp.org

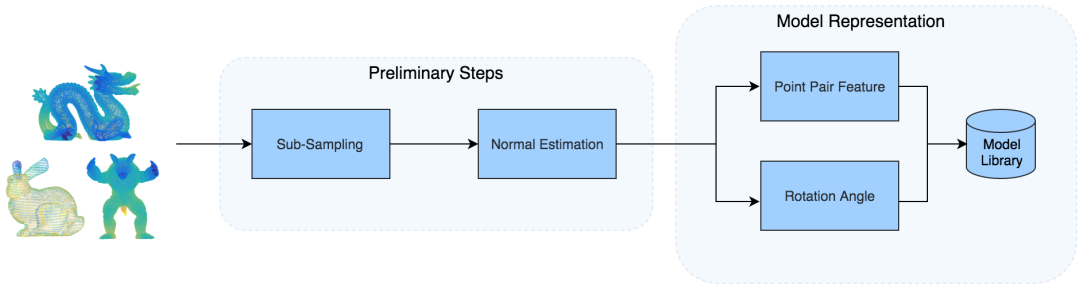

Figure 3.6: Block diagram of PPF off-line phase.

The d is the distance between the two points, as $d = m_2 - m_1$, and the three angles $\in [0; \pi]$. In figure 3.7 is shown the calculation of the F components of two oriented points. F_1 is equal to $\|d\|$, which is the distance of the points. F_2 and F_3 are the angles between the normals and the vectors of the two points. Finally, F_4 is the angle between the two normals.

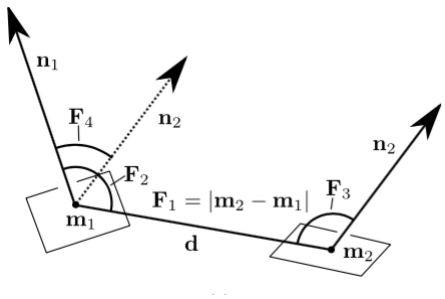

Figure 3.7: Determination of the F vector. Extracted from Drost et al. (2010)

The model library is represented as a hash table, and it is constructed by determining the feature vector for all possible pair of points in the model. The similar features are indexed in the same position in the hash table. Thus, the key to access the library is the feature vector F.

The model rotation angle α_m will only be used in on-line phase to align the object to the input image (also it will be referred as scene), together with the point of the model and the scene rotation angle α_s, it will form the local coordinates. To speed up the on-line phase, the α_m is calculated in this step and saved in the hash table, associated with each point in the model. The mathematical considerations will be described in detail in the next section.

3.4 On-line Phase

The on-line phase is described as a way of processing a scene and choose an appropriate recognition method to find the models present in the scene. Therefore, being an on-line process, this has to be as fast as it can, to minimize the time of recognition.

As the figure 3.8 shows, the first two steps in this diagram are equivalent to the entire off-line phase and are the following: sub-sample scene cloud, normal estimation, determination of the PPF

vector and the scene rotation angle α_s. The last steps consist in finding the possible models to be present in the scene and, lastly, only the correct models among all candidates are selected.

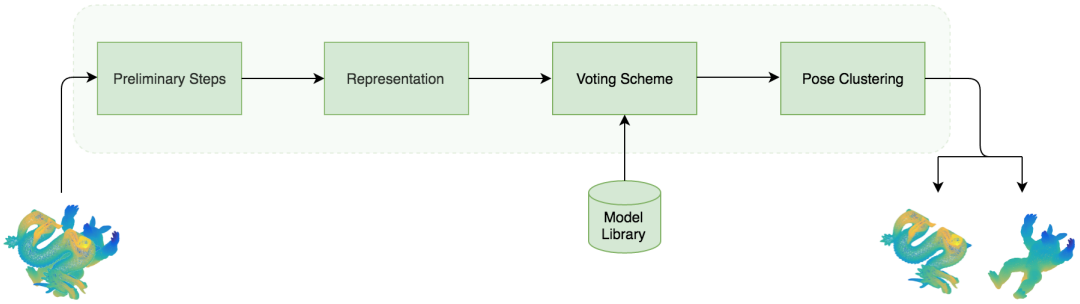

Figure 3.8: Block diagram of PPF on-line phase.

The Voting Scheme is a method of feature matching and hypothesis generation that finds the correspondences between the model features and the scene features. Each match votes for an object pose according to a reference point in the scene. This strategy is related to the Generalized Hough Transform.

For each reference point there is at least one model point that match, this means that both pairs have the same feature F. The search was made using the feature F_s as a key to the hash table of the model library, which returns all similar correspondences.

To construct a valid hypothesis it is essential to transform the model space into the scene space. In other words, rotate the model around the normal of a scene point to align both objects, as the figure 3.9 shows. This movement can be defined by the local coordinates, and that is the pair (m_r, α) based on scene point.

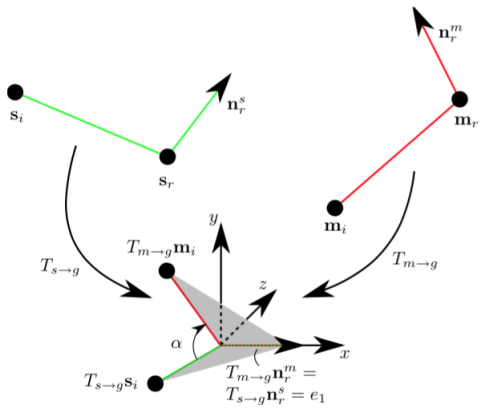

Figure 3.9: Transformation from model space into scene space. Extracted from Drost et al. (2010)

Considering that the scene pair (s_r, s_i) matches with model points (m_r, m_i), the rotation angle α, which is equal to $\alpha = \alpha_m - \alpha_s$, is given by:

$$s_i = T_{s\to g}^{-1} R_x(\alpha) T_{m\to g} m_i \tag{3.5}$$

The $T_{s\to g}$ is the translation of s_r to the origin and the rotation of n_r^s into the x-axis, the $T_{m\to g}$ is equal but for m_r and n_r^m. The $R_x(\alpha)$ is the rotation around x-axis to align s_i and m_i.

This equation can be simplified using $R_x(\alpha) = R_x(-\alpha_s)R_x(\alpha_m)$ and $R_x^{-1}(-\alpha_s) = R_x(\alpha_s)$, to achieve:

$$\begin{cases} t = R_x(\alpha_s)T_{s\to g}s_i \\ R_x(\alpha_m)T_{m\to g}m_i = R_x(\alpha_s)T_{s\to g}s_i \end{cases} \tag{3.6}$$

The arbitrary point $t \in \{\mathbb{R}x + \mathbb{R}_0^+ y\}$, this means that t must belong to the half plane defined by the x-axis where y is positive. The equation 3.6 allows the calculation of α_m and α_s separately and reduces the recognition time in the on-line phase.

After constructing the local coordinates (mr, α), it is necessary to save the votes. To do that, it is created a two-dimensional array, the accumulator array. The number of rows corresponds to the number of points in the model, and the number of columns is the sample steps of α. So, the accumulator vector has to be increased by one in the position $[m_r, \alpha]$, to voting for that local coordinates.

Finally, to maximize the number of model points that coincides with the scene points, each point reference point s_r retrieves a group of potential object poses. These poses are the local coordinates associated with the maximum number of votes and all the votes that are higher than a threshold, in the accumulator array.

Moving further in the block diagram, Drost et al. (2010) uses the Pose Clustering method as hypotheses verification strategy, this rejects the wrong poses and adds accuracy to the final result.

Thus, in this step groups of poses are found, such all the poses in a cluster are within of a rotation and position threshold. In the implementation, various tests were made to find the suitable rotation threshold, where its default value is 12°, and the position threshold has a constant value of 1.5 cm, approximately one tenth of the model's dimension.

Each cluster has a score associated, which is calculated as the sum of the number of votes, estimated in the voting scheme, for each pose in the group. The cluster with the highest score is selected for estimate the final pose; this is achieved by doing the average of all poses in the winner cluster.

3.5 Summary

In this chapter, the problem of the recognition algorithm using PPF was described in detail. Several methods of sub-sampling are simulated to find the most appropriated, the Voxel Grid was the chosen one. After the normal estimation, the PPF are calculated, and the correspondences between the model and the scene are found. Lastly, the hypotheses generation and verification are based on the Generalized Hough Transform and the Pose Clustering.

Chapter 4

Case Study 2: Recognition Algorithm Based On Tri-Spin-Image Features

The other algorithm chosen to solve the problem of the 3D object recognition and to be incorporated in the *Enermeter* tools is based on the TriSI feature proposed by Guo et al. (2015). As mentioned in chapter 2, this method uses three compressed signatures as feature descriptor, and it is efficient under noisy scenes, occlusion, clutter and variations of the mesh resolution.

The authors introduced this method based on their previous work, the Rotation Projection Statistics (RoPS) (Guo et al., 2013). They calculated a local reference frame (LRF) and extracted a descriptor by rotationally projecting the points into 2D planes and calculating a group of statistics. The TriSI outperforms it by improving the LRF with optimal parameters; the feature descriptor concatenates the signatures that express the point distribution in three cylindrical coordinate systems (one for each spin image), rather than rotating the local surface. The feature matching also improves this algorithm by tolerating the adaptation of parameters.

The 3D recognition algorithm proposed has a hierarchical nature, as it is characteristic of the model-based, and it is divided into four steps: off-line processing, feature generation, feature matching and hypotheses verification. Figure 4.1 shows the block diagram of this recognition algorithm. The main processing time and complexity of this method are in the first two stages, where the LRF, the principal component analysis (PCA) and the spin images are calculated.

The algorithm was entirely written in C++, using the PCL and Eigen libraries. In some steps of the implementation, the code available in the PCL library is used, such as the PCA, the LRF construction, and the spin images determination was adapted from the existing ones.

4.1 Off-line Processing and Feature Generation

The off-line phase consists in creating the model database by processing the features points of all 3D models. A sturdy library should have a great amount of models in it; this increases the chance of a possible recognition but increases the time of processing, that is why this step does not belong to the on-line phase.

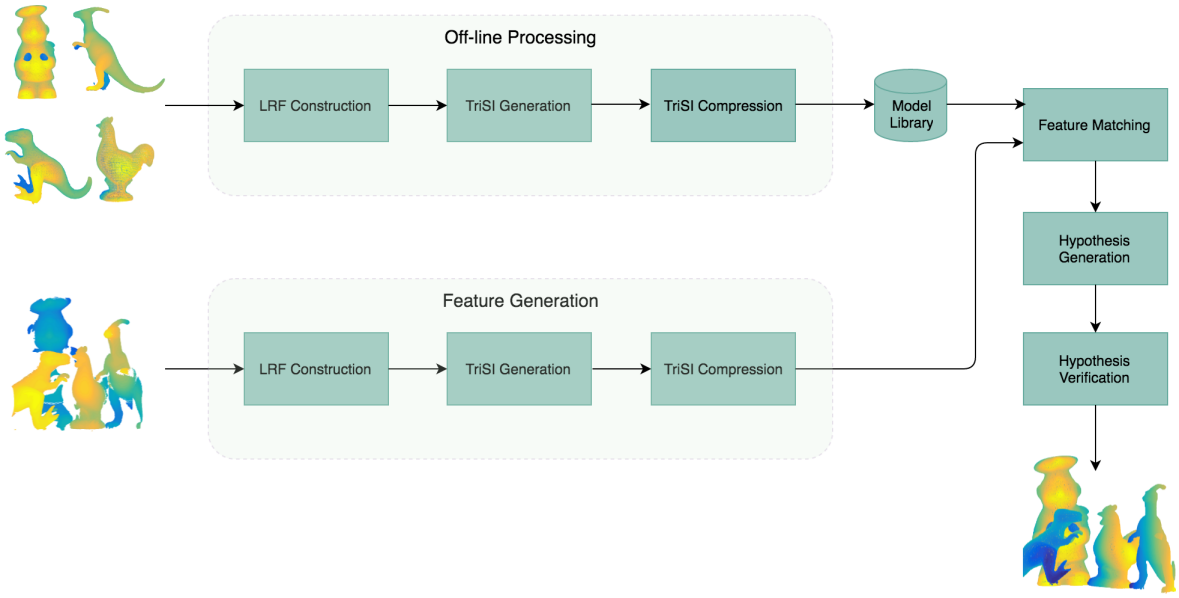

Figure 4.1: Block diagram of the recognition algorithm based on TriSI features.

The on-line stage is characterized by representing the scene and finding a match between it and a model in the library, to identify the location and pose of objects in the scene. To promote the real-time recognition, the representation, and the matching must be very fast, to reduce the time in the on-line phase.

The representation of the models and the scene are the same. Firstly, for each feature, the LRF is obtained and create the TriSI feature by encoding the information from the local surface in three signatures. All features are used to construct the PCA subspace, then each of them is projected into that subspace to obtained the compressed TriSI feature.

4.1.1 LRF Construction

As mentioned before, the local reference frame refers to a coordinate system, which contains the point location in the local surface. The LRF is created by implementing an eigenvalue decomposition on the scatter matrix of all the points in a local surface.

For each selected point, the local surface is calculated by finding the vertices near the point. So, it is assumed that the polygonal information, together with the point cloud, is given as input. If only the point cloud was provided, it is necessary to convert it into a triangular mesh, through a method of triangulation. The figure 4.2 shows an example of a triangular mesh.

Triangulation The triangulation algorithm (Marton et al., 2009), which is implemented in the PCL library, estimates a polygon mesh by projecting the local neighborhoods of a point cloud with its normal vectors associated. The normal vectors can be constructed using the algorithm referred in section 3.2 on the previous chapter.

To calculate the triangular mesh, for a point in the cloud it is necessary to determine the neighborhood, by searching the nearest neighbors in a sphere centered on that point. Next, this neighborhood is projected on a tangent plane to the surface formed by the point and the ones near to it. Finally, the starting triangle is created by finding the two nearest points of the center and connects new triangles for all remaining points.

This method allows the selection by the user of some parameters such as the maximum number of neighbors and the search radius, the appropriate distance between the nearest point and the other points, the maximum and minimum angles of the triangles and the highest angle between the normal vectors.

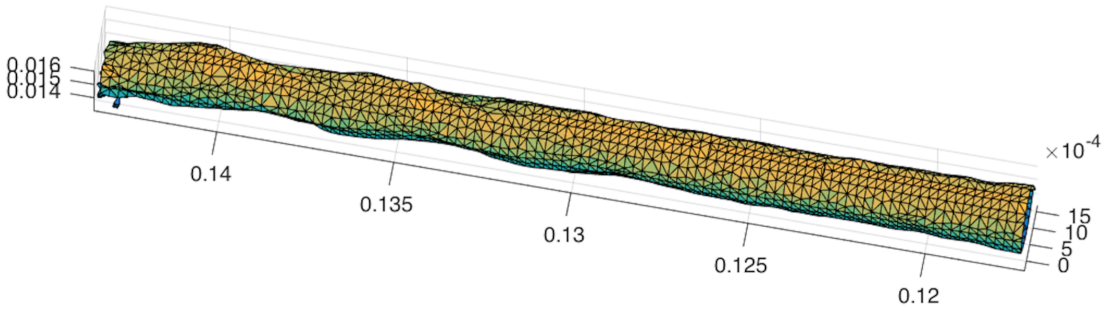

Figure 4.2: Triangular Mesh of the Stanford Drill

After having the triangular information, for a particular point p, the local triangles are calculated such that the vertices in it have a distance r from p. The i-th triangle has three vertices declared as (q_{i1}, q_{i2}, q_{i3}). The scatter matrix of all points in the i-th triangle, S_i, is calculated doing:

$$S_i = \frac{1}{12}\sum_{v=1}^{3}\sum_{u=1}^{3}(q_{iv}-p)(q_{iu}-p)^T + \frac{1}{12}\sum_{v=1}^{3}(q_{iv}-p)(q_{iv}-p)^T \tag{4.1}$$

Therefore, the scatter matrix, S, of all triangles in the local surface is given by:

$$C = \sum_{i=1}^{n_t}\omega_{i1}\,\omega_{i2}S_i \tag{4.2}$$

Where n_t is the number of triangles in the local triangular mesh, ω_{i1} is the ratio between the area of the local triangle, a_i, and the area of the entire local mesh. On the other hand, ω_{i2} is the distance between the centroid of the i-th triangle and the point p. These are obtained by doing:

$$\omega_{i1} = \frac{a_i^{k_1}}{\sum_{i=1}^{n_t} a_i^{k_1}} \qquad \omega_{i2} = (r - \|p - \frac{q_{i1}+q_{i2}+q_{i3}}{3}\|)^{k_2} \tag{4.3}$$

The k_1 is the contribution of a particular triangle related to its surface area and the k_2 is the weight according to the distance to the reference point. The authors simulated several combinations of weights to find the ideal parameters, under the same circumstances. The best values, and

the ones used in the implementation, are $k_1 = 1$ and $k_2 = 2$. However, every combination with $k_1 = 1$ obtained a good performance.

To determine the eigenvalues and the eigenvectors of the scatter matrix, S, the next equation is used, where V is the matrix of the eigenvectors and D is the diagonal matrix of the eigenvalues.

$$V^{-1}SV = D \tag{4.4}$$

Thus, the j-th eigenvalue of the scatter matrix S is obtained doing $D[j,j] = \lambda_j$ and the corresponding eigenvector is the j-th column of the matrix V. The columns of the D and V matrices are in decreasing order of eigenvalues.

The LRF is composed by the three eigenvectores $\{e_1, e_2, e_3\}$ but, due to the sign ambiguity, these $\{-e_1, -e_2, -e_3\}$ are also engeinvectors of the scatter matrix. To resolve this problem, the e_x, $x \in \{1,3\}$, sign can be estimated from sign of the inner product of the eigenvector and the vectors from p to all points q of the local surface. This is achieved doing:

$$\begin{cases} \Lambda_x = sign\left(\sum_{i=1}^{n_t} \omega_{i1}\omega_{i2}\left(\frac{1}{6}\sum_{v=1}^{3}(q_{iv} - p)\, e_x\right)\right) \\ \tilde{e_x} = \Lambda_x e_x \end{cases} \tag{4.5}$$

Where the signum function, $sign()$, returns the value $+1$ or -1 for positive or negative real number, respectively. The eigenvector $\tilde{e_2}$ is given by the cross product of $\tilde{e_1}$ and $\tilde{e_3}$, i.e, $\tilde{e_2} = \tilde{e_1} \times \tilde{e_3}$.

Finally, the LRF feature is completed and the vectors $\{\tilde{e_1}, \tilde{e_2}, \tilde{e_3}\}$ construct the coordinated system centered in the point p, which the x-axis corresponds to $\tilde{e_1}$, y-axis to $\tilde{e_2}$ and z-axis to $\tilde{e_3}$.

The main difference in the calculation of the LRF in this algorithm, when comparing to the RoPS one, is that it had better results in the presence of Gaussian noise, shot noise, and decimation. This improvement is achieved by adding a technique that deals with the irregular triangles. So, the proposed outlier-rejection strategy sets the value of w_1 to 0, if, at least, one edge of the triangle are greater than τ_e times the mesh resolution. The authors suggested the value of τ_e as 5, having in consideration numerous datasets.

4.1.2 Spin Image Determination and TriSI Generation

As described before, the TriSI is based on the Spin Image algorithm of Johnson and Hebert (1998). The spin image method defines a partial cylindrical coordinate system for a point and its normal vector. The descriptor is determined by creating a 2D array that accumulates the computed coordinates for each point of the local surface. This descriptor defines the global shape of the object, and it is constant to rigid transformations.

The 2D accumulator in the TriSI method is formed by saving the point distribution in a cylindric coordinate system and it uses the LRF vector as the reference axis, rather than the normal vector as the Johnson and Hebert (1998) does.

Before going through the mathematical considerations about this algorithm, it is important to define some parameters. The bin size, b_s, is the width of the bins in the accumulator array, and it

corresponds to the mesh resolution, that is the average distance between its vertices. The support distance D_s is the maximum distance between two model points. The image width is the ratio between the support distance and the bin size, i. e., $I_w = \frac{D_s}{b_S}$.

Thus, for a point feature p, its LRF vectors $\tilde{e_x}$, $x \in \{1,2,3\}$, and all the n_p points, q, in its local surface, the coordinates of the spin map is defined by:

$$\alpha = \sqrt{\|q-p\|^2 - (\tilde{e_x} \cdot (q-p)^2)} \qquad \beta = \tilde{e_x} \cdot (q-p) \tag{4.6}$$

Where $\alpha \geq 0$ is the radial coordinate, which is the perpendicular distance of q to the line that passes through p parallel to $\tilde{e_x}$. The elevation coordinate, β, is defined as the signed perpendicular distance to the tangent plane to p and it is perpendicular to $\tilde{e_x}$. The other coordinate of the cylindrical system, the cylindrical angle, can not be defined in a robust way on planar surfaces, so it is excluded.

The next step consists in finding the bin to update the accumulator array SI. The number of rows of this array corresponds to $2I_w + 1$, and the number of columns is the $I_w + 1$. So, the bin point (i, j) is:

$$i = \left\lfloor \frac{D_s - \beta}{b_s} \right\rfloor \qquad j = \left\lfloor \frac{\alpha}{b_s} \right\rfloor \tag{4.7}$$

The $\lfloor w \rfloor$ rounds the w down to the nearest integer. Due to the possible noise in the cloud, it is not sufficient to increment the bin by one. The proposed solution consists in calculating a bilinear interpolation to the four surrounding bins. The bilinear weights are used to increment the bins and are defined as:

$$a = \alpha - ib_s \qquad b = \beta - jb_s \tag{4.8}$$

The bilinear interpolation is achieved doing:

$$\begin{cases} SI(i,j) = SI(i,j) + (1-a)(1-b) \\ SI(i+1,j) = SI(i+1,j) + a(1-b) \\ SI(i,j+1) = SI(i,j+1) + (1-a)b \\ SI(i+1,j+1) = SI(i+1,j+1) + ab \end{cases} \tag{4.9}$$

After processing all feature points of the cloud, the final 2D array, SI, represents the spin image SI_x that encodes the point distribution around the vector $\tilde{e_x}$.

Therefore, the TriSI, f_i, of the point p_i is the three signatures $\{SI_1, SI_2, SI_3\}$ generated by the steps above around the LRF axis $\{\tilde{e_1}, \tilde{e_2}, \tilde{e_3}\}$. These signatures encode the entire local surface and their information are complementary. An example of the tri-spin-images is shown in figure 4.3, where the signatures $\{\tilde{v_1}, \tilde{v_2}, \tilde{v_3}\}$ correspond to $\{\tilde{e_1}, \tilde{e_2}, \tilde{e_3}\}$.

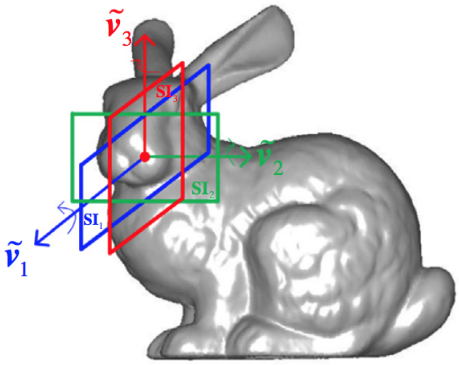

Figure 4.3: Stanford Bunny with the TriSI feature. Extracted from Guo et al. (2015).

4.1.3 TriSI Compression

After the calculation of the TriSi feature and to turn it into a condensed vector, the TriSI characteristic is compressed by project into the principal component analysis subspace.

Firstly, to create the covariance matrix C of the all n_f feature, $\{f_i, ..., f_{n_f}\}$, it is necessary to calculate the average of the tri-spin-images, doing:

$$\bar{f} = \frac{1}{n_f} \sum_{i=1}^{n_f} f_i \tag{4.10}$$

Then, the covariance matrix is obtained using the following equation:

$$C = \sum_{i=1}^{n_f} (f_i - \bar{f})(f_i - \bar{f})^T \tag{4.11}$$

After that, the eigenvalue decomposition is applied to the covariance matrix.

$$V^{-1} C V = D \tag{4.12}$$

Where V is the matrix of the eigenvectors and D is the diagonal matrix of the eigenvalues. The columns of the D and the matrix V are in descending order of eigenvalues.

Finally, the PCA subspace is defined by the transpose of the first column of the matrix, V_0^T, which corresponds to the eigenvector of the highest eigenvalue. So, the compressed feature is given by:

$$\hat{f}_i = V_0^T f_i \tag{4.13}$$

Where $\hat{f}_i$ is the compressed vector of the TriSi feature f_i.

4.2 Feature Matching

As referred in chapter 2, the 3D object recognition algorithm based on TriSI features uses the nearest neighbor in a distance radius for matching the features of the scene to the models in the

library. To reduce the search time, both model and scene TriSI features are saved in a *k-d tree*. This data structure is present in the PCL library, in the *KdTreeFLANN* class.

Therefore, for each scene feature, this strategy finds the nearest and the second nearest distance of a model feature. The model whose feature is the proximal neighbor and the ratio between the first and second nearest features is smaller than a predefined threshold, the correspondence is validated, and the model receives one vote.

Each time a model receives a vote, a rigid transformation is calculated to align the LRF of the model to the LRF of the scene. Considering that there is a feature match $\left(\hat{f}_i^s, \hat{f}_i^m\right)$, the scene LRF matrix of the point p_i^s is $E_i^s = \{\tilde{e_1}, \tilde{e_2}, \tilde{e_3}\}$ and the LRF matrix of the model point p_i^m is E_i^m, the rigid transformation T_i^m can be estimated by:

$$T_i^m = \begin{cases} R = (E_i^s)^T E_i^m \\ t = p_i^s - p_i^m R \end{cases} \tag{4.14}$$

Where R is the rotation matrix and t is the translation vector.

4.3 Hypothesis Generation and Verification

The final step of the recognition algorithm consists in verifying the hypotheses created in the feature matching and extract the correct ones. To do so, the Pose Clustering was the strategy adopted and it consists in finding groups of transformations, where the center of a cluster is a transformation hypothesis.

Therefore, each model has an associated score, which is the number of correspondences achieved in the feature matching. Then, the models are sorted according to their score, and the verification is made individually. A cluster is formed by finding the nearest transformations of the transformation T_i^m, and the center of the cluster presents the final transformation $\hat{T}_i^m$ of the match $\left(\hat{f}_i^s, \hat{f}_i^m\right)$.

Then, each cluster also has a score, based on the number of members, n_c, and their feature distances, d. The clusters with scores smaller than half of the maximum score are rejected from the candidates.

The verification is made individually, for each cluster, the alignment between the model and the scene is performed using the hypothesis transformation. If the model is accurately aligned with a portion of a scene, the model is accepted. Otherwise, the model is rejected and the next candidate is verified.

4.4 Implementation

In this section, some considerations taking into account during the implementation phase of the 3D recognition algorithm will be exposed. For a better understanding, the pseudo code will be presented as a high-level description of the operating principles of the algorithm.

The program was entirely written in C++, using the PCL and Eigen libraries. In some steps of the implementation, the code was adapted to the PCL classes, accurately referred during this exposure.

Starting with LRF construction, the initial step is to calculate the local triangles by finding the vertices near the point, for each feature point, as referred in subsection 4.1.1. The algorithm 1 implements the local triangular mesh.

Algorithm 1 Sergey Ushakov's algorithm for finding all the triangles within the given radius of the given point

Require: The input point *cloud* and the feature *point*
  $vertices \leftarrow getVertices(cloud)$ {Builds the list of triangles for every point of the *cloud*}
  $triangles \leftarrow vertices[point]$
  return Binary search tree *triangles*

After having the local triangular mesh, the LRF matrix is implemented by the algorithm 2. Note that the TriSI method handles with irregular triangles, so this condition had to be performed in the code below.

Algorithm 2 Method that computes LRF matrix for the given point. Adapted from Sergey Ushakov's algorithm.

Require: The input point *cloud*, the feature *point* and the local triangular mesh *triangles*
  $\omega_1[i] \leftarrow getOmega1(point, triangles)$
  $\omega_2[i] \leftarrow getOmega2(point, triangles)$
  for $k = 0$, while $k < 3$, $i{+}{+}$ **do**
    if $triangles.edge[k] > \tau$ **then**
      $\omega_1[i] \leftarrow 0$ {To address irregular triangles}
    end if
  end for
  for $i = 0$, while $i < triangles.size()$, $i{+}{+}$ **do**
    $S[i] \leftarrow getTriangleScatterMatrix(triangle[i], point)$
  end for
  for $i = 0$, while $i < triangles.size()$, $k{+}{+}$ **do**
    $S \leftarrow getScatterMatrix(S[i], \omega_1[i], \omega_2[i])$
  end for
  $eigenVectors \leftarrow computeEigenVectors(S)$
  $lrf \leftarrow signumFunction(point, triangle, eigenVectors, \omega_1[i], \omega_2[i])$
  return Matrix lrf

Then, the TriSI generation is implemented adapting the code of the spin images created by Roman Shapovalov and Alexander Velizhev, in the PCL library.

Finally, the compression of the tri-spin-image feature is shown in the algorithm 4. To obtain eigenvalues and the eigenvectors of the covariance matrix it is necessary to compute the decomposition of this matrix. To do so it is required that the covariance is a square matrix. The dimensions of the new square covariance matrix will be the maximum between the number of rows and the

Algorithm 3 Method that generates the TriSI feature for a given LRF. Adapted from Shapovalov and Velizhev's algorithm.

Require: The input point $cloud$, the feature $point$. the local triangular mesh $triangles$ and the lrf matrix lrf
  for $k = 0$, while $k < lrf.rows()$, $k{+}{+}$ **do**
    for $i = 0$, while $i < triangles.size()$, $i{+}{+}$ **do**
      $q \leftarrow triangles.points[i]$
      $(\alpha, \beta) \leftarrow getSpinMapCoordinates(point, lrf.row(k), q)$
      $(i, j) \leftarrow getBin(\alpha, \beta)$
      $(a, b) \leftarrow getBilinearWeights(\alpha, \beta, i, j)$
      $SI(i, j) \leftarrow SI(i, j) + (1-a)(1-b)$
      $SI(i+1, j) \leftarrow SI(i+1, j) + a(1-b)$
      $SI(i, j+1) \leftarrow SI(i, j+1) + (1-a)b$
      $SI(i+1, j+1) \leftarrow SI(i+1, j+1) + ab$
    end for
  end for
  return 2D Array SI

number columns; the new elements will have the value 0. This way, there is no loss of information during the conversion.

As explained before, the PCA subspace is determined by the transpose of the first column of the eigenvectors matrix, which corresponds to the highest eigenvalue.

Algorithm 4 Method that compresses the TriSI feature fot a given LRF.

Require: The tri-spin-image $feature$
  $avg \leftarrow trisiAverage(feature)$
  $avg \leftarrow convertToSquareMatrix(matrix)$
  $covariance \leftarrow getCovarianceMatrix(avg)$
  $(eigenValues, eigenVectors) \leftarrow eigenvectorsDecomposition(covariance)$
  $triSI \leftarrow compressTriSI(feature, eigenVectors.col[0])$
  return matrix $triSI$

Following the diagram of the figure 4.1, the next procedure is the feature matching. The original process exposed in section 4.2 utilizes a k-d tree as model library, to facilitate the search between model and scene features, using the NNDR strategy to do so. As referred before, this process uses the radius search method included in the *KdTreeFLANN* class. This method only accepts point clouds as input, so it is necessary to create a point cloud of k-d trees, and the PCL library is not prepared to have such structure.

Due to the complexity of the PCL library, this approach was not implemented. The alternative solution consists on saving the model features into an array and finding the nearest neighbors using the NNDR strategy. This method is considerably slower than using the $k-$d tree, since in the worst case makes *O(n)* comparisons and the optimal approach makes *O(log n)*.

Algorithm 5 Method that finds the correspondences between the model and scene features.

Require: The scene point *sceneP*, the model point *modelP*, the scene features *sceneTriSI*, the model features *modelTriSI*, the scene LRF *sceneLRF* and the model LRF *modelLRF*
  for $s = 0$, while $s < sceneTriSI.size()$, $s{+}{+}$ **do**
    for $m = 0$, while $m < modelTriSI.size()$, $m{+}{+}$ **do**
      $near1 \leftarrow getFirstNearest(sceneTriSI[s], modelTriSI[m])$
      $near2 \leftarrow getSecondNearest(sceneTriSI[s], modelTriSI[m])$
      $ratio \leftarrow getRatio(near1, near2)$
    end for
  end for
  if $ratio < \tau$ **then**
    $modelVotes[m] \leftarrow modelVotes[m] + 1$
    $transformation \leftarrow alignLRF(sceneP, modelP, sceneLRF, modelLRF)$
  end if
  return Array *modelVotes* and Eigen::Affine $transformation$

Finally, the last step is the hypothesis generation and verification. In this stage, the clusters are created by finding a group of neighboring transformations. Then, the final transformation will be the average of the transformations of the cluster. The rejection of the false hypotheses consists in excluding the groups with scores lowest than half the maximum score; then the accepted models will be rotated and aligned to the scene.

Algorithm 6 Method that creates the clusters and rejects the false correspondences.

Require: The array *modelVotes*, the vector of allrigid transformation $transf$ and the model *model*
  $modelVotes \leftarrow sort(modelVotes)$
  $maxVote \leftarrow getMax(modelVotes)$
  for $v = 0$, while $v < modelVotes.size()$, $v{+}{+}$ **do**
    $tranformation_{avg} \leftarrow 0$
    if $modelVotes[m] < maxVote/2$ **then**
      $cluster.vote \leftarrow modelVotes[m]$
      for $t = 0$, while $t < transf.size()$, $t{+}{+}$ **do**
        $tranformation_{avg} \leftarrow tranformation_{avg} + transf[t]$
      end for
      $tranformation_{avg} \leftarrow tranformation_{avg}/transf.size()$
      $cluster.transf \leftarrow tranformation_{avg}$
    end if
  end for
  $outCloud \leftarrow getTransformPointCloud(model, cluster.transf)$
  return The transformed point cloud *outCloud*

4.5 Summary

In this chapter, the problem of the recognition algorithm using TriSI feature was described in detail. The off-line process and the generation of the scene features are equal and divided into

the LRF construction, TriSI generation, and compression. The LRF of the RoPS method does not support irregular triangles and the TriSI one handles it. Lastly, the hypotheses generation and verification are based on the Pose Clustering strategy.

The chapter finishes with the considerations made during the implementation step, as well as the pseudo code of the most important stages of this 3D object recognition algorithm.

Chapter 5

Experiments and Results

In this chapter, the results of the recognition methods based on PPF and based on TriSI features will be presented. Firstly, the values of the parameters that achieved the best results will be studied, for both methods. In the next two sections, several experiments in synthetic datasets and in real ones will be described, such as the addition of Gaussian noise, variations of mesh resolution and occluded objects.

To compare the performance of the two algorithms were used two standard datasets and one provided by Enermeter, which is used in their projects. The standard models used were from the Stanford 3D scanning repository [1] and the YCB model set Çalli et al. (2015), used in robotic manipulation.

As referred before, the algorithms are implemented in C++, compatible with the version 1.8 of the PCL library. The experiments were run in a Windows 8 OS, in an Intel Core i7-4770K @ 3.4GHz with 4GB RAM.

5.1 Selection of Parameters

In this section, the parameters of PPF and TriSI based methods are trained using a simple scene and few models in the library. The models used were the Happy Buddha, the Dragon, and the Stanford Bunny, from the Stanford repository, as shown in the figures 5.1, 5.2 and 5.3. The scene was created from the Stanford Bunny by downsampling it to 1530 points, reducing about 1/23 of its original number of points, this scene is shown in figure 3.2 of chapter 3.

5.1.1 PPF Algorithm

In the point pair feature algorithm, to find the suitable values of the sampling rate τ_d and the rotation threshold θ_r, several experiments were made using the default values. For determining the sampling rate, it was used 1.5 cm for position threshold and 12° for rotation threshold. On the other hand, the value used for sampling rate was 0.05 and the same amount for position threshold.

[1] http://graphics.stanford.edu/data/3Dscanrep/

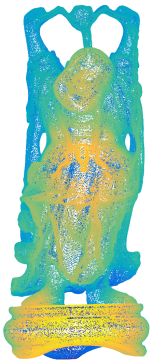

Figure 5.1: Model of the Happy Buddha

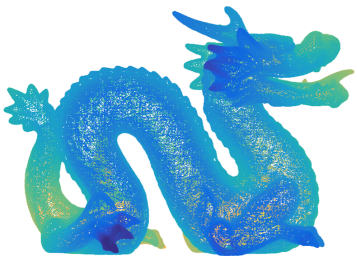

Figure 5.2: Model of the Dragon.

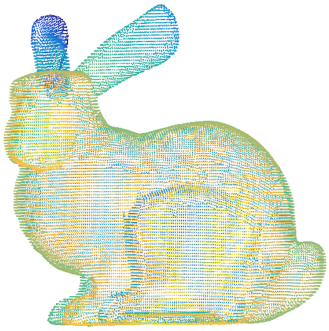

Figure 5.3: Model of the Stanford Bunny.

The τ_d is used in the on-line phase to calculate the scene reference point to be utilized in the determination of the rigid movement between the model and the input image. To test the influence of this parameter, the recognition rate was measured according to the value of the sampling rate, this started with 1 and grew up to 10, with a step of 1. Figure 5.4 shows the results of this experiment and to obtain reliable conclusions, the algorithm was executed three times for each value.

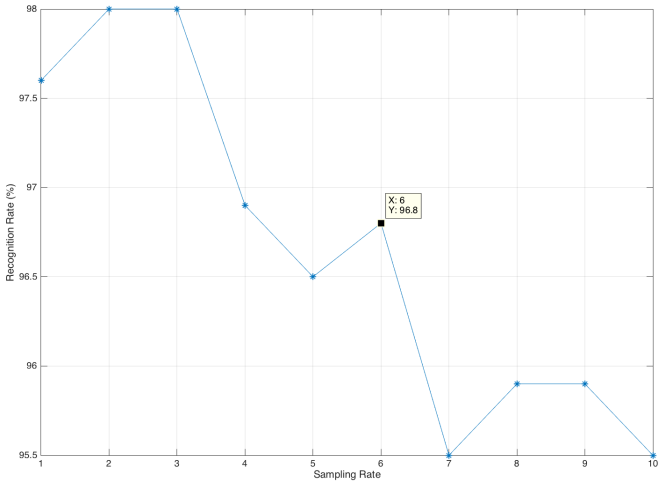

Figure 5.4: Analysis of the optimal value of the sampling rate τ_d for PPF algorithm.

Analyzing the figure 5.4, the obvious conclusion is that for lowest numbers of the sampling rate the recognition rate increases, however, the run-time rises significantly. As referred before, this parameter is the step in the index of the scene points, for instance, for a $\tau_d = 1$, it will be considering all the points to construct the scene PPF and takes about 20 min to finish the program. And for $\tau_d = 10$, only 1/10 will be utilized for that calculations and the program runs for 3 min.

Therefore, it is necessary to find a symbiotic relationship between the recognition rate (RR) and running time. Since the input scene is equal to a subsampled model, it is predictable that the recognition ratio has highest values, it will be choose a value above 96 % and a sampling rate

below 4. So, the desirable τ_d has 6 with a RR of 96.8 %.

The θ_r is used for constructing the clusters after a rigid transformation is calculated. This parameter ensures that in one cluster the retrieved pose does not differ in rotation for more than θ_r. To examine the influence of this parameter, it was measured the recognition rate according to several values of the rotation threshold, from 0° and to 20°, with a step of 2.5°. The sampling rate used was the one obtained above. Figure 5.5 shows the results of this experiment.

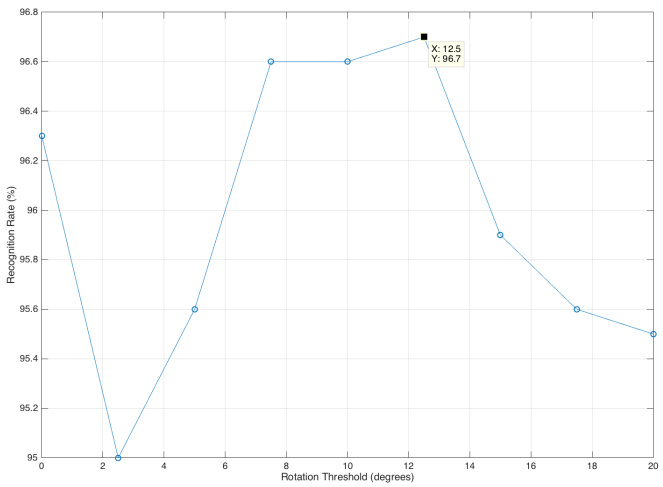

Figure 5.5: Results of the recognition rates against the rotation threshold.

Since the changing of this parameter does not affect the recognition time, it will be selected the one with the highest score of the recognition rate. As the figure 5.4 shows, the peak of the recognition rate occurs at 96.7 % for a rotation threshold of 12.5°.

5.1.2 TriSI Algorithm

In the TriSI algorithm, some points in the cloud will be selected to compute the features and execute the feature matching and the hypothesis generation and verification. Since the algorithm gets the point neighboring information locally, this does not affect the global performance of this method. So, for each model it selects N points, indices, randomly.

Several tests were made to find the proper number of indices to be applied in the recognition algorithm, using the same models and scene of the previous experiments. This parameter started with the value 10 and grew up to 1000. Figure 5.6 shows the results of this evaluation, where the local surface is estimated in a radius of 1 cm. To achieve solid conclusions, the algorithm was executed three times for each value.

Since this algorithm calculates the local information around each indice, for highest values of N the run-time increases drastically, making the results not available in useful time. For instance, for three models in the library and $N = 500$, the program finishes after 15 min running, which for large datasets it will be inconceivable.

Due to its computational cost, it will be choose a value lower than 500, but with the best recognition rate. So, it was selected the value $N = 100$, for at RR of 81 %.

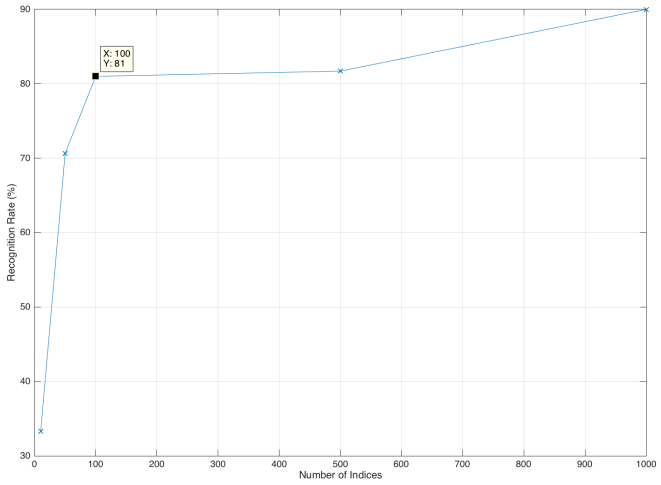

Figure 5.6: Recognition rate using different numbers of indices.

The other parameter estimated is the threshold of the feature matching process. After finding the first and the second nearest distances between a model and a scene, if the ratio of the distances is small enough, the object feature will be considered. So, for a high threshold, the recognition accuracy is increased, as well as the computational cost.

To study the influence of the parameter it was measured the recognition rate according to several values of the τ, from 0.6 to 1, with a step of 0.1. The used number of indices was the one obtained above. Figure 5.7 shows the results of this research.

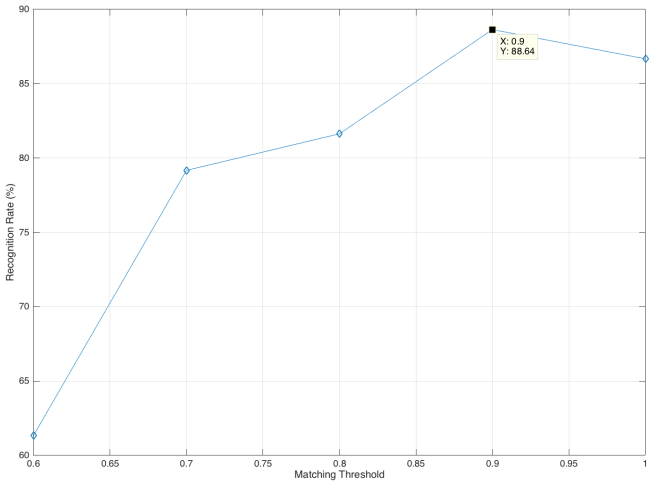

Figure 5.7: Analysis of the suitable value for the matching threshold τ.

For the feature matching threshold, it will be selected the one with the highest score of the recognition rate. As the figure shows, the greatest value of the recognition rate occurs at 88.64 % for a τ of 0.9.

5.2 Synthetic Dataset

As mentioned before, to analyze the performance of the both 3D model-based recognition algorithms extensive experiments were realized in a synthetic database.

The YCB is a larger dataset composed of 88 objects divided into five categories: food items, kitchen items, tool items, shape items, and task items. Figures 5.8 and 5.9 show some of the 20 objects used in this experiments. The models were constructed using a system composed by 5 RGBD sensors and 5 high-resolution RGB cameras arranged in a quarter-circular arc, and it can be accessed in the form of point clouds or polygonal meshes.

Figure 5.8: Tool items of Çalli et al. (2015) dataset.

Figure 5.9: Food items of Çalli et al. (2015) dataset.

5.2.1 Additive Gaussian Noise

The first experiment consists of evaluating the performance of the recognition algorithm with additive Gaussian noise applied to a scene of the YCB dataset. Three levels of Gaussian noise were added, with standard deviations between 1 % to 5 % of the mesh resolution, *mr*. The results are exposed in table 5.1 for the YCB models.

Algorithm	**Gaussian Noise σ (%mr)**		
	1	**3**	**5**
PPF	69.6%	62.4%	58.4%
TriSI	75.8%	60.6%	59.5%

Table 5.1: Recognition Rate in the presence of Gaussian Noise on the YCB models.

As the table suggest, as long as the deviation of the Gaussian noise increase, the recognition rates decrease. The PPF and TriSI algorithms achieved comparable results at high levels of Gaussian noise, between 3 %*mr* and 5 %*mr*. However, the recognition rate of the TriSI feature was the best result to a low level of noise, with a RR of 75.8 % to a standard deviation of 1 % of the mesh resolution.

5.2.2 Variation of the Mesh Resolution

In the next step of the comparative study, the different recognition rates are registered according to variations of the mesh resolution. The scene was created by subsampling a model cloud in $1/2$, $1/3$ and $1/4$ of its initial dimension. The table 5.2 shows the results for a large dataset.

Algorithm	**Mesh Resolution**		
	1/2	**1/3**	**1/4**
PPF	69.8%	72.1%	69.6%
TriSI	64.3%	45.6%	35.6%

Table 5.2: Influence of the mesh resolution variation in the RR of the YCB database.

As the table shows, the recognition levels of the PPF algorithm are approximately constants under $1/2$ to $1/4$ mesh decimation, with the highest RR of 72.1 %. The TriSI algorithm achieved a high recognition rate of 64.3 % and it dropped significantly as long as the mesh decimation increase.

It was expected that TriSI based algorithm would not be so sensitive to mesh resolution because the three spin images are calculated using a LRF rather than a normal vector, which makes a discriminative feature, encoding more information of the local surface.

5.3 Real Dataset

To finalize the experiments on 3D object recognition, a dataset provided by Enermeter was used, which is composed of five models, three types of metal bonding, a metal nut and a stopper, and a cluttered scene. The figure 5.10 shows an example of a point cloud belonging to its database. The process of 3D modeling has not been entirely implemented since the 3D acquisition was obtained only with a single viewpoint. It is necessary to overlap multiple views to cover the complete object's surface, then the view correspondences are found and aligned in a coordinate system. The final model represents a smooth surface of the free-form object, and it allows its visualization from all sides.

Algorithm	**Model**				
	Cork Stopper	**Bonding Type I**	**Bonding Type II**	**Bonding Type III**	**Metal Nut**
PPF	100%	100%	94.5%	100%	39.8%

Table 5.3: Recognition rates of the Enermeter dataset usins the PPF approach.

Only the results of the algorithm based on PPF features will be presented, the recognition rate of the TriSI approach showed inconclusive values to this dataset. A possible justification can be the increase of the model self-occlusion, since it is represented with a single viewpoint.

The table 5.3 shows the recognition rates against a cluttered scene, represented in figure 5.11, using the algorithm based on PPF feature. This algorithm achieved an average RR of 87.2 % for all of the five objects, the highest value for an individual object was 100 % and the lowest was of 39.8 %. So, the PPF algorithm achieved satisfactory results even in conditions of lack of descriptiveness in the model representation.

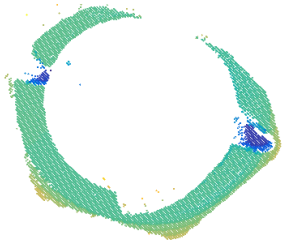

Figure 5.10: Point cloud of the metallic nut.

Figure 5.11: Cluttered scene of the Enermeter's dataset.

5.4 Summary

In this chapter, several experiments are executed to evaluate the performance of the recognition algorithms based on PPF and TriSI features. Initially, the ideal parameters for both methods were studied using a simple scene and a model library with few objects.

Then, tests were made in a synthetic dataset in conditions of Gaussian Noise and variation of the mesh resolution, using the values discovered above. The TriSI feature obtained the best results for small levels of Gaussian Noise, but the PPF one has the most stable rates for the decimation of the mesh resolution.

Finally, the algorithms are assessed utilizing the Enermeter's databases. The results of PPF algorithm were satisfactory, with an average of RR of approximately 90 %. However, the TriSI features achieve inconclusive results for this dataset.

Unfortunately, due to the limited computational resources, it was not possible to evaluate the algorithms under more rigorous experiments, such as the comparison between a small and an extensive model library, more levels of the low standard deviation of the Gaussian Noise and other different types of noise and tests in occluded and cluttered synthetic scenes.

Chapter 6

Conclusion and Future Work

The 3D data is becoming more affordable, due to the recent development of 3D imaging sensors and the real-time simultaneous localization and mapping techniques. Therefore, the technological advance of the artificial vision field was increasing exponentially in the past decades, as well as the necessity of developing autonomous systems capable of acquired real-time information and act according to its purpose.

Enermeter, a local company that works with computer vision systems that inspect and optimize quality standards, proposed the selection and implementation of a suitable 3D recognition algorithm to be incorporated in their engines. The biggest challenge is that the strategy needs to be compatible with the Point Cloud Library (PCL), which is a C++ complicated library that can process point clouds and 2D/3D images.

The first logic step was to study the different approaches about algorithms of 3D recognition, to select strategies to develop and produce a detailed evaluation. After being aware of the recent advancements in this area it was elected an algorithm based on point pair features and other based on the traditional spin-images, the tri-spin-images features.

The recognition algorithm based on point pair features consists in creating a descriptor that represents the point cloud globally, although the recognition was the particularity of being achieved locally, grouping and aligning similar characteristics. The first phase of this method is the subsampling of the initial point cloud, using the Voxel Grid class, then it is necessary to calculate the normal vector to obtain the PPF feature. The Generalized Hough Transform and the Pose Clustering methods are used to generate and verify the transformation hypotheses. All the stages of this approach were entirely implemented using the code available in the PCL.

On the other hand, the strategy based on tri-spin-images creates three features that encode the geometric information around all points of the entire cloud, and each model is verified individually to select the one that best characterize the object. This problem was divided into four stages: offline processing, feature creation, hypotheses generation, and verification. The first two are equal and composed by the construction of the LRF, the generation, and compression of the TriSI feature. The algorithm was entirely written in C++, and some steps of the implementation were used the available code in the PCL library.

Finally, to evaluate the referred proposals, several experiments were executed using two synthetic databases and one with models related to the industrial environment in which the Enermeter works. After studying the ideal parameters for both methods, the TriSI algorithm obtained the best results for small levels of Gaussian Noise, but the PPF feature has the most stable rates for mesh resolution variations. The TriSI features achieved inconclusive results for the Enermeter's dataset, although, the results of PPF algorithm was adequate.

Concerning the comparative study, it was chosen the recognition algorithm based on point pair features to be capable of being incorporated into the existing Enermeter's tools. This conclusion was the logic one since the implemented method based on TriSI does not reveal to be appropriate to its purpose.

The improvement of the TriSI algorithm was impossible to achieve in an useful time, due to the limited computational resources. However, the results were considered positive since the comparison between the two algorithms was achieved, and one strategy was selected.

6.1 Future Work

A list of additional features that can be implemented in the future is presented bellow:

- The integration of a suitable feature detection technique in both algorithms to extract distinctive features of the model surface.

- The implementation of the model library of the TriSI algorithm using a *k*-d tree, using the *KdTreeFLANN* class of the PCL library.

- The application of a suitable search method, such as the nearest neighboring distance radius, using *k*-d trees, in the feature matching of the TriSI algorithm.

- Increase the complexity of the hypotheses verification process, by verifying the alignment individually with a portion of a scene, until all transformations are checked or too few remain in the last scene portion.

- Find the proper values of the rotation and translation thresholds to be applied in Pose Clustering, to verify the alignment of the model and the scene.

- The comparison between the hypotheses generation and verification using Pose Clustering and RANSAC technique, for both algorithms.

- Increase the number of experiments used to evaluate the algorithms performance, adding low levels of Gaussian, Laplacian, and short noise, implemented in a robust real dataset and larger synthetic one.

- The inclusion of more techniques in the comparative study, some traditional algorithms as the spin-images and RoPS one and a modern approach, like the Volumetric and Multi-View Convolutional Neural Networks.

References

Çalli, B., Walsman, A., Singh, A., Srinivasa, S., Abbeel, P., and Dollar, A. M. (2015). Benchmarking in manipulation research: The YCB object and model set and benchmarking protocols. *CoRR*, abs/1502.03143. Available from: http://arxiv.org/abs/1502.03143.

Campbell, R. J. and Flynn, P. J. (2001). A Survey Of Free-Form Object Representation and Recognition Techniques. *Computer Vision and Image Understanding*, 81(2):166–210. Available from: http://www.sciencedirect.com/science/article/pii/S1077314200908890.

Chua, C. S. and Jarvis, R. (1997). Point Signatures: A New Representation for 3d Object Recognition. *International Journal of Computer Vision*, 25(1):63–85. Available from: http://link.springer.com/article/10.1023/A%3A1007981719186.

Drost, B., Ulrich, M., Navab, N., and Ilic, S. (2010). Model globally, match locally: Efficient and robust 3d object recognition. In *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 998–1005.

Elgammal, A. (2010). CS 534: Computer Vision 3d Model-based recognition. Available from: http://www.cs.rutgers.edu/~elgammal/classes/cs534/lectures/3D_modelbasedvision.pdf.

Guo, Y., Bennamoun, M., Sohel, F., Lu, M., and Wan, J. (2014). 3d Object Recognition in Cluttered Scenes with Local Surface Features: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2270–2287.

Guo, Y., Sohel, F., Bennamoun, M., Lu, M., and Wan, J. (2013). Rotational Projection Statistics for 3d Local Surface Description and Object Recognition. *International Journal of Computer Vision*, 105(1):63–86. Available from: http://link.springer.com/article/10.1007/s11263-013-0627-y.

Guo, Y., Sohel, F., Bennamoun, M., Wan, J., and Lu, M. (2015). A novel local surface feature for 3d object recognition under clutter and occlusion. *Information Sciences*, 293:196–213. Available from: http://www.sciencedirect.com/science/article/pii/S0020025514009219.

Ho, H. T. and Gibbins, D. (2008). Multi-scale feature extraction for 3d surface registration using local shape variation. In *Image and Vision Computing New Zealand, 2008. IVCNZ 2008. 23rd International Conference*, pages 1–6.

Hu, J. and Hua, J. (2009). Salient spectral geometric features for shape matching and retrieval. *The Visual Computer*, 25(5-7):667–675. Available from: http://link.springer.com/article/10.1007/s00371-009-0340-6.

Johnson, A. E. and Hebert, M. (1998). Surface matching for object recognition in complex three-dimensional scenes. *Image and Vision Computing*, 16(9–10):635–651. Available from: http://www.sciencedirect.com/science/article/pii/S0262885698000742.

Knopp, J., Prasad, M., Willems, G., Timofte, R., and Gool, L. V. (2010). Hough Transform and 3d SURF for Robust Three Dimensional Classification. In Daniilidis, K., Maragos, P., and Paragios, N., editors, *Computer Vision – ECCV 2010*, number 6316 in Lecture Notes in Computer Science, pages 589–602. Springer Berlin Heidelberg. DOI: 10.1007/978-3-642-15567-3_43. Available from: http://link.springer.com/chapter/10.1007/978-3-642-15567-3_43.

Marton, Z. C., Rusu, R. B., and Beetz, M. (2009). On fast surface reconstruction methods for large and noisy point clouds. In *IEEE International Conference on Robotics and Automation, 2009. ICRA '09*, pages 3218–3223.

Mian, A., Bennamoun, M., and Owens, R. (2005). 3d model-based free-form object recognition – a review.

Mian, A., Bennamoun, M., and Owens, R. (2006). Three-Dimensional Model-Based Object Recognition and Segmentation in Cluttered Scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1584–1601.

Mikolajczyk, K. and Schmid, C. (2005). A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630.

Pope, A. (2004). Model-Based Object Recognition. Technical report. Available from: ftp://ftp-admin.cs.ubc.ca/.snapshot/hourly.0/local/techreports/1994/TR-94-04.pdf.

Qi, C. R., Su, H., Nießner, M., Dai, A., Yan, M., and Guibas, L. (2016). Volumetric and multi-view cnns for object classification on 3d data. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*.

Sipiran, I. and Bustos, B. (2011). Harris 3d: a robust extension of the Harris operator for interest point detection on 3d meshes. *The Visual Computer*, 27(11):963–976. Available from: http://link.springer.com/article/10.1007/s00371-011-0610-y.

Sun, J., Ovsjanikov, M., and Guibas, L. (2009). A Concise and Provably Informative Multi-Scale Signature Based on Heat Diffusion. *Computer Graphics Forum*, 28(5):1383–1392. Available from: http://onlinelibrary.wiley.com/doi/10.1111/j.1467-8659.2009.01515.x/abstract.

Unnikrishnan, R. and Hebert, M. (2008). Multi-scale interest regions from unorganized point clouds. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08*, pages 1–8.

Zhong, Y. (2009). Intrinsic shape signatures: A shape descriptor for 3d object recognition. In *2009 IEEE 12th International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 689–696.